SERVICING A COMPONENT-BASED SOFTWARE PRODUCT THROUGHOUT THE SOFTWARE PRODUCT LIFECYCLE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is a continuation-in-part of co-pending U.S. patent application Ser. No. 09/561,389, filed Apr. 27, 2000, entitled "A COMPONENTIZED OPERATING SYSTEM," which is hereby incorporated herein by reference in its entirety for all purposes.

TECHNICAL FIELD

[0002] Embodiments of the present invention relate to the field of servicing software products. In particular, embodiments of this invention relate to updating component-based operating systems and application programs with a service package throughout the software product lifecycle.

BACKGROUND OF THE INVENTION

[0003] An operating system image interacts with hardware in a computer to perform a multitude of functions. Conventional computers generally execute the operating system to manage various aspects of the computer as it is running. Typically, the operating system is responsible for managing access to storage devices as well as input and/or output devices, and controlling the execution of one or more additional applications. Before the operating system may be executed by the computer, it typically must be installed on the computer, a process which usually involves copying multiple files from a distribution medium (e.g., a CDROM) onto a storage device (e.g., a hard disk) of the computer.

[0004] A typical operating system includes a large number of files which may include instructions and/or data. These instructions, when executed by the computer, provide the operating system functionality. The operating system may be modified (e.g., updated) in any of a wide variety of manners, such as by adding or replacing one or more particular files, by any of a wide variety of people (e.g., a user, administrator, software developer other than the operating system developer, etc.). It becomes difficult to troubleshoot a malfunctioning computer or update the operating system because it is difficult for the user or administrator to know exactly what functionality is or should be installed on the computer.

[0005] In existing systems, servicing the binary files that comprise a software product (e.g., the operating system or an application program) is often the most expensive aspect of a software product lifecycle. An exemplary software product lifecycle includes a pre-deployment phase (e.g., pre-installation), a deployment phase (e.g., installation), and a post-deployment phase (e.g., on the running system). The size of the binary files, the amount of binary files that are typically serviced for any single problem, and the different locations of the binary files during each of the software product lifecycle phases make the distribution and creation of the binary files difficult. Also, servicing multiple binaries across the system fails to provide specific information on the current version of any larger aggregated piece of functionality on the system.

[0006] For example, when creating a service package to remedy an issue in software code stored in a binary file, existing systems typically create an updated copy of the binary or a patch that modifies the software code without modifying other software. However, existing systems typically require different versions of both the patches and the binary files based on the current phase of the software product lifecycle: one version for pre-deployment, one version for during deployment, and one version for the running system.

[0007] In another example, some prior systems provide a "hot" fix (e.g., a patch) for the end user that replaces a single file or library (e.g., library.dll) on a running system or during predeployment. Other systems provide updates during installation by directing the client machine to obtain the updated file or library (e.g., library.dll) from an installation medium or via a network and install the obtained update. However, such systems require separate scripts for each lifecycle phase of the client machine: pre-deployment, deployment, and post-deployment. Further, a change to a single file may necessitate a change to other dependent files. The prior systems fail to provide for intelligent dependency resolution during the update process.

[0008] Accordingly, a system for servicing a software product across the entire software product lifecycle is desired to address one or more of these and other disadvantages.

SUMMARY OF THE INVENTION

[0009] Embodiments of the invention include updating a software product with a service package. The service package includes one or more files associated with the component and a plurality of instruction sets for installing the files. In an embodiment, the invention determines a state or other operating context associated with the component, selects one of the instruction sets based on the determined state, and applies one or more of the files to the component in accordance with the selected instruction set. For example, the state may correspond to one of the following phases of a software product lifecycle: predeployment of the software product, deployment of the software product, and post-deployment of the software product.

[0010] The invention provides a single service package to both consumers and original equipment manufacturers to service a software product during all parts of the software product lifecycle. As software products have more and more binaries to service, the invention reduces the cost of servicing and enables a simple query of the system to determine the version of binaries serviced on the system.

[0011] In accordance with one aspect of the invention, a computerized method updates a software product. The method includes defining the software product as a plurality of components. The method also includes accessing one of a plurality of instruction sets associated with a service package based on a lifecycle phase of the software product to be updated. The method also includes modifying at least one of the components in the software product in accordance with the accessed instruction set.

[0012] In accordance with another aspect of the invention, a method updates a software product with a service package. The software product includes a plurality of components. Each of the components has a state associated therewith. The state represents an operating context of the component. The